STUDIA INFORMATICA 2010
Volume 31 Number 2B (90)

Anna PIWOWARCZYK-CYBULA, Adam PIÓRKOWSKI
Akademia Górniczo-Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej

WYKORZYSTANIE PRZESZUKIWANIA PEŁNOTEKSTOWEGO W KONSTRUKCJI APLIKACJI O ARCHITEKTURZE CIENKIEGO KLIENTA[1]

Streszczenie. Architektura cienkiego klienta jest obecnie powszechnie stosowanym rozwiązaniem w tworzeniu aplikacji do zarządzania przedsiębiorstwem. Wraz ze wzrostem liczby użytkowników systemu przed konstruktorami stają problemy obciążenia aplikacji. W przypadku dziedzin, w których interakcja użytkownika wiąże się z licznymi wyszukiwaniami, celowe jest zapewnienie optymalnych operacji selekcji. Wyszukiwanie może odbywać się także w rozbudowanych opisach tekstowych dotyczących magazynowanych encji. Wówczas warto jest sięgnąć po indeksy pełnotekstowe. Niniejszy rozdział przedstawia analizę realizacji tego mechanizmu w różnych systemach zarządzania bazami danych. Przykłady dotyczą struktury i zapytań rzeczywistej bazy danych dla rynku nieruchomości.

Słowa kluczowe: baza danych, przeszukiwanie pełnotekstowe, ranking

USE FULL-TEXT SEARCH IN CONSTRUCTION APPLICATION OF THIN CLIENT ARCHITECTURE

Summary. Thin client architecture is now widely used solution for developing applications for business management. With the increase in users of the system architects are problems loading the application. In areas where user interaction is associated with many searches, it is appropriate to ensure optimal operation of selection. Searches may also take place in a complex textual descriptions of the stored entities. Then we turn to the full-text indexes. This article presents an analysis of the implementation of this mechanism in different database management systems. Examples include the structure and query the actual database of real estate market.

Keywords: database, full-text search, ranking

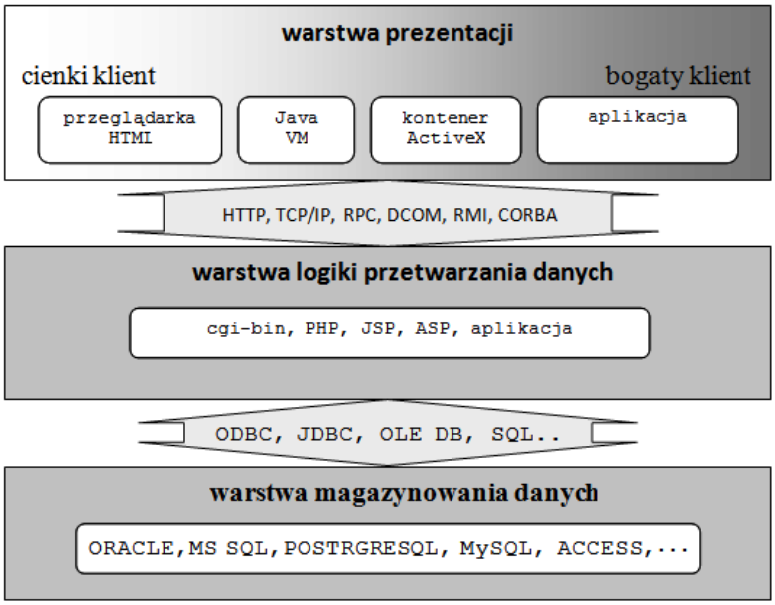
[1] Praca finansowana w ramach badań statutowych KGIS nr 11.11.140.561.

1. Wprowadzenie

Obserwując na przestrzeni ostatniej dekady rozwój różnego rodzaju oprogramowania można zaobserwować pewną tendencję do centralizacji przetwarzania danych przy jednoczesnym upraszczaniu oprogramowania klienckiego. Najwyraźniejszym tego przykładem w ostatnich latach mogą być coraz popularniejsze usługi przetwarzania w chmurze (ang. *Cloud Computing*), które coraz mocniej promują w swojej ofercie tacy potentaci, jak Google, Microsoft, Yahoo. Innym przykładem może być polska firma tworząca oprogramowanie dla biur nieruchomości, której produkty ewoluowały od aplikacji jednostanowiskowych poprzez architekturę klient-serwer w całości instalowaną w biurze nieruchomości, aż po model scentralizowany, w którym aplikacja administracji ofertami nieruchomości udostępniana jest w całości jako usługa zdalna (ang. *SaaS*, *Software as a Service*)[1].

W przedstawionych przykładach rolę cienkiego klienta pełni dowolna przeglądarka internetowa. Taki model tworzenia oprogramowania jest wygodny z punktu widzenia dostawcy, ponieważ pozwala na scentralizowane zarządzanie i aktualizację, nie wymagając tym samym żadnych zmian po stronie końcowego użytkownika w jego środowisku klienckim.

Na rysunku 1 przedstawiono typowy trójwarstwowy model tworzenia aplikacji z zaznaczeniem różnicy pomiędzy cienkim klientem (ang. *thin client*) a bogatym klientem (ang. *rich client*).



Rys. 1. Model trójwarstwowy
Fig. 1. Three-tier architecture

Tematem niniejszego artykułu jest jedno z zagadnień wykorzystywanych w wielu różnorodnych aplikacjach, w tym również w zarządzaniu ofertami nieruchomości, tj. przeszukiwanie pełnotekstowe realizowane w warstwie magazynowania danych. Na podstawie udostępnionych autorom danych z tej branży opracowano testy wydajności i porównanie ich wyników dla kilku systemów baz danych.

2. Indeksowanie pełnotekstowe

Indeks pełnotekstowy (ang. *full text index*) jest specjalnego rodzaju indeksem opartym na słowach kluczowych, który przyspiesza wyszukiwanie w bazach danych. Dzięki stosowaniu indeksów pełnotekstowych uzyskujemy dostęp do funkcji umożliwiających rozszerzanie lub zawężanie kryteriów wyszukiwania.

Na proces tworzenia indeksu pełnotekstowego składa się:

- filtrowanie treści dokumentu tekstowego, czyli usunięcie formatowania oraz metadanych, np. w przypadku dokumentów XML lub HTML,
- parsowanie dokumentu,
- usuwanie nieznaczących słów, takich jak np. przyimki,
- operacja sprowadzenia słów do ich formy podstawowej,
- wyszukiwanie synonimów oraz słów pochodnych.

Wyszukiwanie pełnotekstowe pozwala na wyszukiwanie danych tekstowych na podstawie ich zgodności z pojedynczymi słowami, frazami czy różnymi formami danego słowa [2]. Dzięki temu można tworzyć bardziej skomplikowane warunki wyszukiwania niż za pomocą operatora `LIKE`. Zaletą wyszukiwania pełnotekstowego jest łatwość pobierania fragmentów danych zapisanych w pojedynczych kolumnach. Dodatkowo, gdy z bazą danych łączy się jednocześnie wielu użytkowników, stosowanie indeksów pełnotekstowych może stać się koniecznością

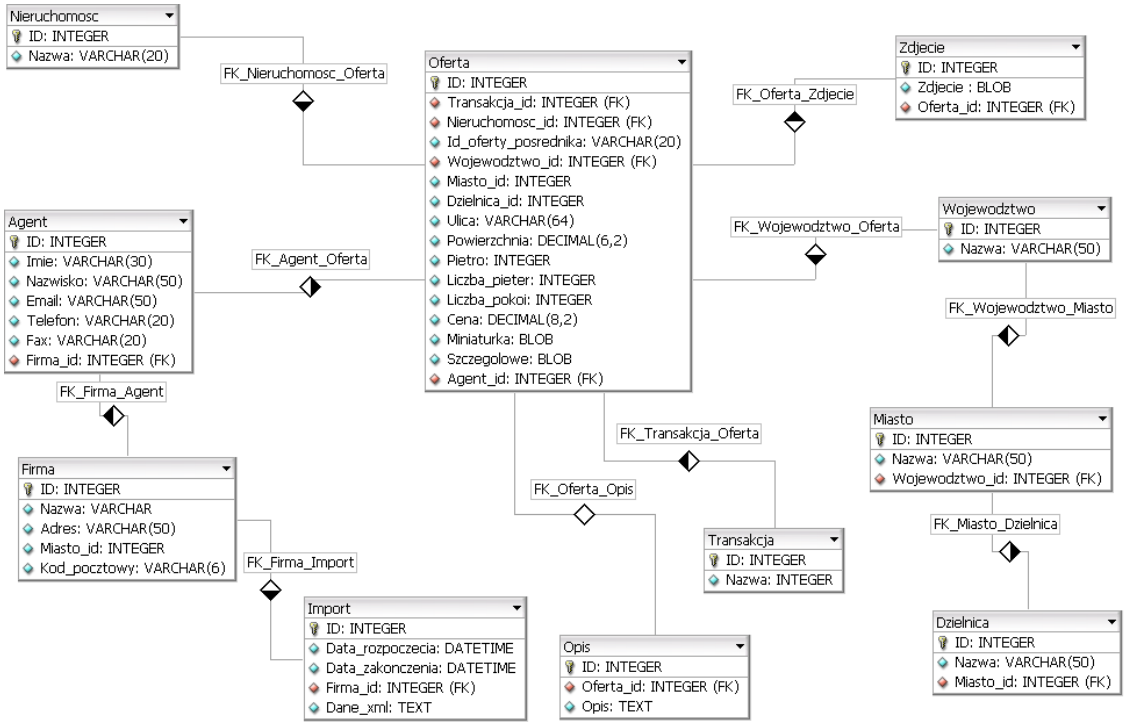
Każdy z porównywanych serwerów oferuje wyszukiwanie pełnotekstowe z uwzględnieniem tzw. rankingu wyników. Ranking wyników pozwala na określenie trafności uzyskanych rezultatów wyszukiwania poprzez nadanie odpowiedniej wagi, na przykład waga 1.0 dla wyniku spełniającego warunki zapytania w całości lub waga 0.0 dla wyniku nie spełniającego żadnych kryteriów. Oczywiście, możliwe są również wartości pośrednie.

Dla każdego serwera możliwe jest również utworzenie i modyfikacja zestawu często używanych słów, które nie podlegają indeksowaniu (ang. *stopword file*).

3. Indeksacja w systemach zarządzania bazą danych

Dla każdego z wykorzystanych do porównania serwerów baz danych została stworzona baza danych o dokładnie takim samym schemacie. Diagram (rys. 2) przedstawia schemat bazy dla serwera MySQL.

Do indeksowania i przeszukiwania pełnotekstowego wykorzystano kolumnę *Opis* tabeli *Opis*, gdyż zawierała dużą ilość rekordów o znacznych rozmiarach danych tekstowych.



Rys. 2. Schemat bazy danych dla serwera MySQL 5.1
Fig. 2. The database schema for MySQL 5.1

3.1. Indeksacja i przeszukiwanie pełnotekstowe w MySQL 5.1

W serwerze MySQL indeksy pełnotekstowe mogą zostać stworzone jedynie dla tabel typu MyISAM, co powoduje, iż klucze obce wiążące tabele są ignorowane [3]. Takie indeksy mogą zostać stworzone dla kolumn przechowujących dane typu `CHAR`, `VARCHAR` oraz `TEXT`. Tworzenie indeksu może odbywać się zarówno podczas operacji tworzenia tabel, jak i później, poprzez użycie `CREATE FULLTEXT INDEX` lub `ALTER TABLE`.

Wyszukiwanie pełnotekstowe odbywa się poprzez wywołanie zapytania `SELECT` z klauzulą `WHERE MATCH(col1, col2,…) … AGAINST(wzorzec [tryb])`, gdzie klauzula `MATCH()` zawiera listę kolumn (wymienione po przecinku), w których odbywa się przeszukiwanie. Klauzula `AGAINST()` zawiera natomiast określenie, jakie łańcuchy znakowe będą wyszukiwane. Ważne jest, iż wyszukiwana wartość ujęta w klauzuli `AGAINST()` nie może być zmienną lub nazwą kolumny. W klauzuli tej za pomocą odpowiedniego modyfikatora określa się również typ przeszukiwania pełnotekstowego.

Serwer MySQL oferuje następujące trzy typy przeszukiwania pełnotekstowego:

- `IN BOOLEAN MODE` – tryb ten pozwala na stosowanie różnego rodzaju operatorów, jak na przykład + lub -, które pozwalają na lepsze określenie poszukiwanego wzorca. Przy przeszukiwaniu w tym trybie nie jest konieczne tworzenie indeksu pełnotekstowego,
- `IN NATURAL LANGUAGE` – tryb ten jest dostępny, gdy w klauzuli `AGAINST()` podany zostanie jedynie wzorzec. Pozwala on na określenie tzw. wartości semantycznej wyraże-

nia. Przykładowo, im wzorzec częściej występuje w wierszach przeszukiwanych kolumnach, tym wartość semantyczna takiego wiersza jest większa. W wynikach zapytania wyświetlane są jedynie wiersze o niezerowej wartości semantycznej. Stosowanie operatorów jak w przypadku poprzedniego trybu nie przynosi efektów,

- `WITH QUERY EXPANSION` – to tak zwane wyszukiwanie z rozwijaniem zapytania, które jest modyfikacją trybu `IN NATURAL LANGUAGE`. W trybie tym tabela/tabele są przeszukiwane dwa razy. Pierwsze przeszukanie odbywa się w trybie `IN NATURAL LANGUAGE`, następnie przy drugim wyszukiwaniu wzorzec zostaje połączony z kilkoma innymi wzorcami, mającymi największą wartość semantyczną pierwotnego wzorca.

Określenie stopnia trafności uzyskanych wyników, czyli tzw. ranking wyników, można uzyskać jedynie poprzez wyszukiwanie w trybie `IN NATURAL LANGUAGE`.

3.2. Indeksacja i przeszukiwanie pełnotekstowe w PostgreSQL 8.3

Serwer PostgreSQL umożliwia, podobnie jak MySQL, realizowanie wyszukiwania pełnotekstowego bez konieczności tworzenia indeksów pełnotekstowych, jednakże są one wskazane, gdy przeszukiwanie pełnotekstowe odbywa się często [4]. PostgreSQL dysponuje dwoma typami indeksów pełnotekstowych, a mianowicie:

- GiST (ang. *Generalized Search Tree*)-based index – kolumny, w których odbywa się przeszukiwanie, muszą być typu `tsvector` lub `tsquery`. Ten typ indeksu ma pewną wadę, ponieważ może generować fałszywe wyniki,
- GIN (ang. *Generalized Inverted Index*)-based index – kolumny, w których odbywa się przeszukiwanie, mogą być jedynie typu `tsvector`.

`tsvector` to typ danych służący przechowywaniu przetworzonego dokumentu. Można opisać go jako posortowaną listę leksemów, czyli wyrazów będących abstrakcyjnymi jednostkami systemu słownikowego. Typ `tsquery` jest natomiast używany dla zapytań tekstowych, umożliwiając stosowanie operatorów logicznych w celu lepszego określenia poszukiwanego wzorca.

Tworzenie indeksu pełnotekstowego odbywa się poprzez wywołanie:

- `CREATE INDEX name ON table USING gist(column)` – dla indeksów typu GiST,
- `CREATE INDEX name ON table USING gin(column)` – dla indeksów typu GIN.

Wyszukiwanie pełnotekstowe realizowane przy wykorzystaniu indeksów GIN jest około trzy razy szybsze niż przy korzystaniu z indeksów GiST.

Serwer PostgreSQL umożliwia tworzenie rankingu trafności wyników. Realizowane jest to za pomocą funkcji `ts_rank` lub `ts_rank_cd`. Obie mają taką samą składnię, natomiast różnica polega na tym, iż druga z nich wykorzystuje funkcję gęstości pokrycia (ang. *cover density*), opisaną w artykule [5].

3.3. Indeksacja i przeszukiwanie pełnotekstowe w SQL Server 2008

Serwer MS SQL Server 2008 [6] przed utworzeniem indeksu pełnotekstowego (`CREATE FULLTEXT INDEX`) wymaga utworzenia katalogu pełnotekstowego (`CREATE FULLTEXT CATALOG`), w którym przechowywane są indeksy pełnotekstowe. Istnieje możliwość stworzenia takiego katalogu dla każdego indeksu oddzielnie. Podczas tworzenia indeksu pełnotekstowego dostępne są różnorodne opcje wskazujące, kiedy indeks ma być uaktualniony (`MANUAL`, `AUTO`, `OFF [,NO POPULATION]`). Uaktualnienie bądź wypełnienie indeksu odbywa się poprzez wywołanie funkcji `ALTER FULLTEXTINDEX`. Indeks pełnotekstowy może zostać utworzony dla kolumn typu `CHAR`, `VARCHAR`, `NCHAR`, `NVARCHAR`, `TEXT`, `NTEXT`, `IMAGE`, `XML`, `VARBINARY`.

Wyszukiwanie pełnotekstowe odbywa się poprzez wywołanie zapytania `SELECT` z klauzulą `CONTAINS` lub `FREETEXT`, co odpowiada wyszukiwaniu pełnotekstowemu bez rankingu wyników. Wyszukiwanie z rankingiem wyników jest realizowane przez funkcje `CONTAINSTABLE` oraz `FREETEXTTABLE`. Serwer umożliwia wyszukiwanie zarówno pojedynczych słów lub fraz, jak i słów będących synonimami danego wzorca. Możliwe jest stosowanie różnego rodzaju operatorów do zapisu wzorca.

3.4. Indeksacja i przeszukiwanie pełnotekstowe w IBM DB2 9.5

Serwer IBM DB2 [7] wymaga przed utworzeniem indeksu pełnotekstowego wywołania procedury systemowej (`SYSPROC.SYSTS_ENABLE`), która umożliwia stworzenie takiego indeksu. Tworzenie indeksu również odbywa się poprzez wywołanie procedury systemowej (`SYSPROC.SYSTS_CREATE`), w przeciwieństwie do pozostałych serwerów baz danych, gdzie operacja ta odbywa się poprzez wywołanie funkcji SQL.

Wyszukiwaną frazę umieszcza się w klauzuli `CONTAINS(kolumna, 'wzorzec')`, znajdującej się w części określającej warunek zapytania `SELECT`. Możliwe jest stosowanie operatorów logicznych w celu dokładniejszego określenia szukanej frazy. Aby ustalić trafność zapytania, serwer udostępnia funkcję `SCORE(kolumna, 'wzorzec')`.

3.5. Indeksacja i przeszukiwanie pełnotekstowe w Oracle 11g

W przypadku serwera Oracle 11g tworzenie indeksów pełnotekstowych (`CREATE INDEX`) odbywa się bez konieczności wcześniejszego przygotowania bazy danych [8]. Wyszukiwaną frazę podobnie jak w przypadku serwera IBM DB2 umieszcza się w klauzuli `CONTAINS(kolumna, 'wzorzec')`. W celu określenia trafności zapytania serwer udostępnia operator `SCORE(etykieta)` powiązany z numerem etykiety frazy umieszczonej w klauzuli `CONTAINS`, na przykład:

```
SELECT *, SCORE(label) FROM tab WHERE CONTAINS(tab, 'wzorzec', label);
```

Tak jak w przypadku wszystkich pozostałych serwerów baz danych do, określenia wzorca wyszukiwania możliwe jest użycie operatorów logicznych.

4. Testy wydajności zapytań

4.1. Warunki testów

Do porównania wydajności zostały wybrane najnowsze wersje popularnych serwerów baz danych, a były to: MySQL 5.1, PostgreSQL 8.3, Oracle 11g Release 2 Enterprise Edition, IBM DB2 9.5 Express-C, SQL Server 2008 Express Edition.

Bazy danych utworzone na poszczególnych serwerach nie były optymalizowane w żaden sposób.

Dla każdego z serwerów została przygotowana pula procedur i funkcji składowanych, odpowiadających za manipulowanie danymi, które następnie podczas odpowiednich testów wydajności były wywoływane poprzez krótki program napisany w języku Java. Do połączenia z wybranym serwerem baz danych zostały użyte sterowniki JDBC (ang. *Java Database Connectivity*) udostępnione przez producentów serwerów baz danych.

Środowisko testowe stanowiły serwer i klient o następujących parametrach:

- serwer: procesor AMD Athlon X2 4600 (2.4 GHz), pamięć RAM 4 GB DDR2 800 MHz, dysk twardy Seagate Barracuda 320 GB 7200.10 (cache 16 MB, Serial ATA II),
- klient: procesor Intel Core Duo (1.6 GHz), pamięć RAM 1.5 GB DDR 667 MHz, dysk twardy Toshiba 60 GB 5400.

W związku z faktem, iż sieć, do której należały klient oraz serwer, cechowała się dużą zmiennością przepustowości, komputery te na czas testów połączono ze sobą bezpośrednio kablem krzyżowym (ang. *crossover*), tak aby wykorzystać maksymalną przepustowość łącza (100 Mbps) oraz wyeliminować wpływ innych czynników na testy wydajności. W czasie testów zarówno klient, jak i serwer, nie były obciążane innymi zadaniami, a na serwerze był uruchomiony tylko jeden wybrany serwer baz danych.

W celu porównania wpływu systemu operacyjnego zainstalowanego na serwerze testy wydajności odbywały się z wykorzystaniem systemu operacyjnego Linux (Ubuntu 8.10) oraz Windows 2008 Server. W czasie testów system Linux był uruchomiony w trybie konsolowym.

Podczas testów wydajności przeszukiwania pełnotekstowego wyszukiwano oferty spełniające następujące warunki:

- zawierające słowo *„mieszkanie”* (zwane dalej *zapytaniem 1*)

```
SELECT * FROM "Opis" WHERE CONTAINS("Opis", 'mieszkanie') > 0;
```

- zawierające słowa *„mieszkanie”* oraz *„umeblowane”*, ale nie zawierające frazy *„ciemna kuchnia”* (zwane dalej *zapytaniem 2*),

```
SELECT * FROM "Opis"
WHERE CONTAINS("Opis", 'mieszkanie and umeblowane -{ciemna kuchnia}') > 0;
```

- zawierające frazę *„luksusowe mieszkanie”*, ale bez wystąpienia wyrażenia *„ciemna kuchnia”* (zwane dalej *zapytaniem 3*),

```
SELECT * FROM "Opis"
WHERE CONTAINS("Opis", '{luksusowe mieszkanie} - {ciemna kuchnia}') > 0;
```

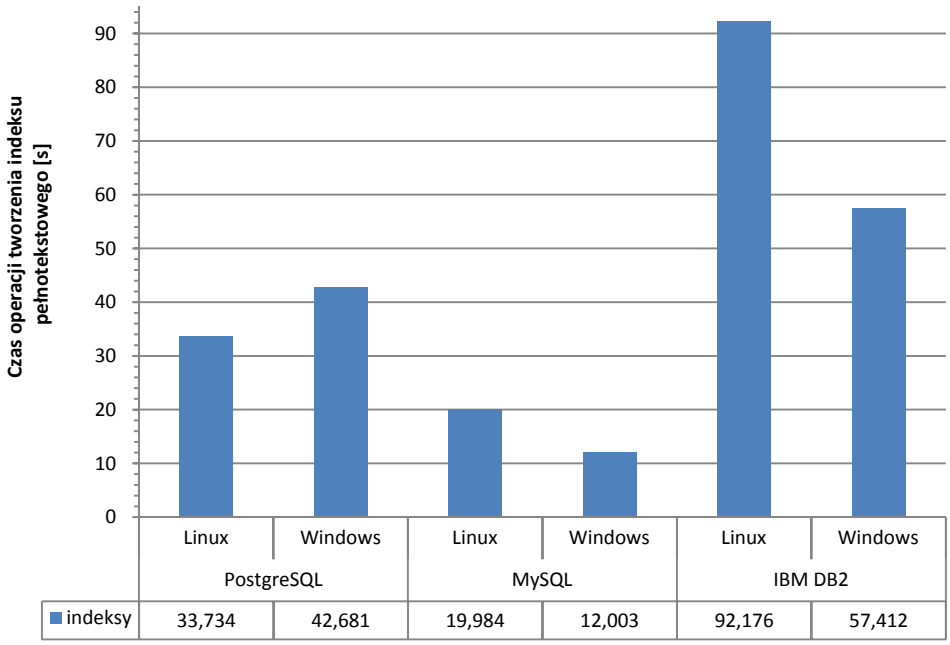
- zawierające słowa *„mieszkanie”*, *„luksusowe”*, *„salon”*, a także opcjonalne słowo *„nowoczesne”* oraz nie zawierające frazy *„aneks kuchenny”* (zwane dalej *zapytaniem 4*),

```
SELECT * FROM "Opis" WHERE
CONTAINS("Opis", 'mieszkanie and luksusowe and salon - {aneks kuchenny} no-
woczesne');
```

- zawierające słowa *„mieszkanie”*, *„umeblowane”*, frazę *„dobry dojazd”*, ale nie zawierające frazy *„aneks kuchenny”* (zwane dalej *zapytaniem 5*).

```
SELECT * FROM "Opis" WHERE
CONTAINS("Opis", 'mieszkanie and umeblowane and (dobry dojazd)
- {aneks kuchenny}') > 0;
```

Kod SQL dla przeszukiwania pełnotekstowego przedstawiono dla bazy danych Oracle, dla pozostałych baz za wyjątkiem PostgreSQL był on bardzo podobny.



Rys. 3. Zestawienie średnich czasów tworzenia indeksów pełnotekstowych na kolumnie z opisem tabeli "Opis" dla systemów Windows oraz Linux

Fig. 3. Summary of average time to create full-text index on a column with a description of the table "Description" for Windows and Linux

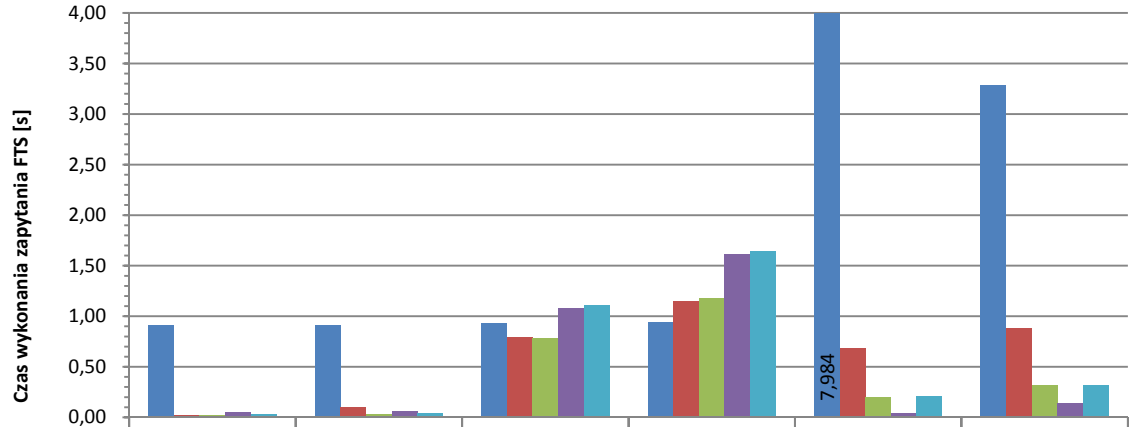
4.1. Wyniki testów

Ze względu na ograniczenia licencyjne serwerów Oracle oraz SQL Server 2008, w tej części artykułu mogą zostać przedstawione jedynie wyniki uzyskane dla serwerów MySQL, PostgreSQL oraz IBM DB2.

4.1.1. *Indeksowanie pełnotekstowe*

Rysunek 3 prezentuje zbiorcze wyniki przeprowadzonych testów wydajności operacji tworzenia indeksu pełnotekstowego Najwolniej tworzenie indeksu przebiegało dla serwera IBM DB2. Czas poświęcony na wykonanie operacji, dla serwera uruchomionego pod kontrolą systemu Linux, był ponad 4,5-krotnie dłuższy niż dla serwera MySQL. Operacja przebiegała szybciej na serwerze IBM uruchomionym pod kontrolą systemu Windows, jednakże wydłużenie czasu w stosunku do serwera MySQL nie było mniejsze.

4.1.2. *Przeszukiwanie pełnotekstowe bez rankingu wyników*



	PostgreSQL Linux	PostgreSQL Windows	MySQL Linux	MySQL Windows	IBM DB2 Linux	IBM DB2 Windows
zapytanie 1	0,909	0,914	0,934	0,938	7,984	3,284
zapytanie 2	0,019	0,105	0,797	1,145	0,686	0,881
zapytanie 3	0,017	0,036	0,781	1,177	0,205	0,317
zapytanie 4	0,047	0,058	1,079	1,613	0,041	0,145
zapytanie 5	0,036	0,041	1,109	1,649	0,209	0,322

Rys. 4. Zbiorcze zestawienie średnich czasów wykonania zapytań FTS bez rankingu wyników dla systemów operacyjnych Linux i Windows

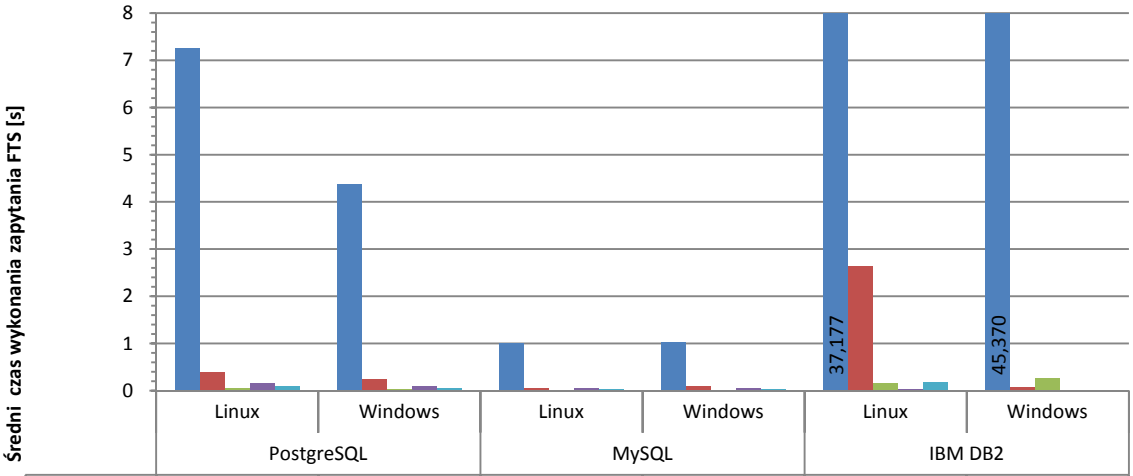
Fig. 4. Aggregate average execution times of queries without the ranking of FTS results for Linux and Windows operating systems

Rysunek 4 przedstawia zbiorcze zestawienie czasów wykonania zapytań pełnotekstowych dla poszczególnych serwerów baz danych, a tabela pod wykresem zawiera dokładne czasy wykonania poszczególnych zapytań.

W wyniku wykonania pierwszego zapytania zwróconych zostało najwięcej rekordów, dlatego też czas wykonania tego zapytania i pobrania wyników dla wszystkich serwerów jest

najdłuższy. Warto zauważyć, iż serwery niekomercyjne, PostgreSQL oraz MySQL, osiągnęły w tym przypadku czasy krótsze niż komercyjny serwer firmy IBM. Dla serwera MySQL zaobserwowano, iż stopień skomplikowania warunków zapytania ma większy wpływ na czas zwrócenia wyników niż ilość zwróconych rezultatów (czas potrzebny na otrzymanie rezultatu działania rósł dla kolejnych zapytań, mimo iż ilość zwracanych wyników była, w stosunku do rezultatów z pierwszego zapytania, znacznie mniejsza).

4.1.3. *Przeszukiwanie pełnotekstowe wraz z rankingiem wyników*



	PostgreSQL		MySQL		IBM DB2	
	Linux	Windows	Linux	Windows	Linux	Windows
zapytanie 1	7,256	4,370	1,016	1,039	37,177	45,370
zapytanie 2	0,395	0,242	0,046	0,095	2,638	0,072
zapytanie 3	0,055	0,031	0,016	0,019	0,155	0,272
zapytanie 4	0,164	0,098	0,047	0,047	0,034	0,022
zapytanie 5	0,089	0,052	0,031	0,033	0,189	0,022

Rys. 5. Zestawienie średnich czasów wykonania zapytań FTS z rankingiem wyników dla serwerów pod systemami Linux i Windows

Fig. 5. Aggregate average execution times of queries with the ranking of FTS results for Linux and Windows operating systems

Rysunek 5 przedstawia zbiorcze zestawienie czasów wykonania zapytań pełnotekstowych wraz z rankingiem wyników dla poszczególnych serwerów baz danych. Wyniki zostały posortowane od najbardziej do najmniej trafnego. Tabela pod wykresem przedstawia dokładny czas wykonania poszczególnych zapytań w sekundach.

Analiza otrzymanych wyników wykazała wzrost czasu potrzebnego na wykonanie wszystkich zapytań dla serwera PostgreSQL, w stosunku do zapytań bez rankingu wyników. Dla serwerów MySQL oraz IBM DB2 zaobserwowano wzrost czasu podczas wykonywania pierwszego zapytania. Dla serwera MySQL wzrost ten był niewielki, natomiast dla IBM DB2 wersja uruchomiona pod kontrolą systemu Linux była ponad 4,5-krotnie wolniejsza, a wersja pod kontrolą systemu Windows prawie 14-krotnie mniej wydajna. Można sądzić, iż znaczący wpływ w tej sytuacji miała operacja sortowania dużej liczby wyników według rankingu.

5. Wnioski

W wyniku przeprowadzonych testów najbardziej wydajnym serwerem baz danych okazał się serwer Oracle, niestety, zapisy licencyjne nie pozwoliły na prezentację wyników. Wśród serwerów niekomercyjnych godnym uwagi okazał się serwer MySQL, który w stosunku do serwera PostgreSQL osiągał lepsze wyniki podczas operacji tworzenia indeksu pełnotekstowego oraz wyszukiwania pełnotekstowego wraz z rankingiem wyników. Serwer MySQL pozwala również, jako jedyny wśród pozostałych serwerów, na wyszukiwanie pełnotekstowe bez konieczności tworzenia indeksu pełnotekstowego. Mankamentem takiego rozwiązania jest jednakże brak możliwości powiązania rezultatów wyszukiwania z ich trafnością w stosunku do kryteriów wyszukiwania.

Najgorsze rezultaty osiągnięte zostały przez serwer IBM DB2. Czasy operacji indeksowania i przeszukiwania pełnotekstowego bez rankingu wyników były w jego przypadku najdłuższe. Serwer zachowywał się również wysoce niewydajnie podczas operacji przeszukiwania pełnotekstowego z rankingiem wyników w sytuacji, gdy zwracana była bardzo duża liczba wyników. Osiągnięte wyniki wskazują, iż najbardziej kosztowna w takiej sytuacji była operacja sortowania wyników od najbardziej do najmniej trafnych.

Dla serwera IBM DB2 w wyniku przeprowadzonych testów stwierdzono, iż operacja stworzenia indeksu pełnotekstowego nie jest równoznaczna z jego wypełnieniem. W celu odczytania i przetworzenia danych należy dodatkowo wywołać procedurę systemową.

Dla serwera MySQL należy pamiętać, iż indeksowanie i wyszukiwanie pełnotekstowe jest możliwe jedynie na tabelach typu MyISAM, dla których zachodzi cache'owanie indeksów. Oznacza to, iż zapytanie pełnotekstowe, które będzie często wykonywane, będzie szybciej zwracało wyniki przy kolejnym wywołaniu.

BIBLIOGRAFIA

1. Henderson K.: Bazy danych w architekturze klient/serwer. Wydawnictwo Robomatic, Wrocław 1999.
2. Cieślewicz J., Pelikant A.: Reprezentacja i wyszukiwanie dokumentów tekstowych w bazach danych. Studia Informatica, Vol. 30, 2A, Politechnika Śląska, 2009.
3. MySQL 5.1 Reference Manual, http://dev.mysql.com/doc/refman/5.1/en/
4. PostgreSQL 8.3.6 Documentation, http://www.postgresql.org
5. Clarke C. L. A., Cormack G. V., Tudhope E. A.: Relevance Ranking for One to Three Term Queries. Information Processing and Management , 36(2):291-311, 2000.
6. SQL Server 2008 Product Documentation, http://msdn.microsoft.com

7. Centrum informacyjne produktu IBM DB2, http://publib.boulder.ibm.com/info-center/db2luw/v9r5/index.jsp
8. Oracle Database Documentation Library, http://www.oracle.com/pls/db111/homepage

Recenzenci: Dr inż. Adam Duszeńko
Dr inż. Hafed Zghidi

Wpłynęło do Redakcji 12 stycznia 2010 r.

Abstract

The theme of this article is one of the issues used in many different applications which is full-text search implemented in data storage layer. In order to enable full-text searches most database servers require a full-text index to be created. The only exception is MySQL server. It is not necessary when you use searching mode `IN BOOLEAN MODE`. All servers also allow you to sort the search results using their relevancy to the search terms. For some servers, such as IBM DB2 or SQL Server 2008, it is necessary to advance the preparation of a database to enable full-text searching. Usually it is implemented by calling system functions or using SQL. Also the operation of creation the full-text index to some servers is not equivalent with its full filling. This is true in the case of IBM DB2 and SQL Server 2008. In order to read and process the data you must additionally call the procedure or function of the SQL system.

Fig. 3 summarizes the average time to create full-text index on a text column named "*Opis*" for Windows and Linux. As a result of the experiment, in both cases, the best results were obtained for the MySQL server. Fig. 4 presents the results achieved for FTS queries without ranking the results for the operating systems Linux and Windows while Fig. 5 shows summary of average execution times of queries with ranking results. In this test the best results were obtained also for the MySQL server.

Adresy

Anna PIWOWARCZYK-CYBULA: piwowarczyk.cybula@gmail.com
Adam PIÓRKOWSKI: Akademia Górniczo-Hutnicza, Katedra Geoinformatyki i Informatyki Stosowanej, al. Mickiewicza 30, 30-059 Kraków, Polska, pioro@agh.edu.pl